PENJELASAN DARI KODE PEMBELAJARAN

PERTEMUAN 5

Nama : Muuhammad Luqmqnul Fikri

NIM : 231011400546

Kelas : 05TPLE017

□ 1. Persiapan Data

```python
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv("processed_kelulusan.csv")
X = df.drop("Lulus", axis=1)
y = df["Lulus"]
```

◈ Membaca dataset processed_kelulusan.csv.
◈ Memisahkan fitur (X) dan target (y), di mana kolom Lulus adalah variabel target yang ingin diprediksi.

```python
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, random_state=42)
```

◈ Dataset dibagi menjadi:

- 70% data latih (train)
- 15% validasi (val)
- 15% pengujian (test)
  Pembagian **stratify=y** memastikan proporsi kelas tetap seimbang.

⚙️ 2. Pipeline dan Preprocessing

```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
```

🔹 Mengimpor komponen preprocessing: imputasi (isi nilai kosong) dan standarisasi fitur numerik.

```
num_cols = X_train.select_dtypes(include="number").columns
```

🔹 Menentukan kolom numerik yang akan diproses.

```
pre = ColumnTransformer([
    ("num", Pipeline([("imp", SimpleImputer(strategy="median")),
                      ("sc", StandardScaler())]), num_cols),
], remainder="drop")
```

🔹 ColumnTransformer membuat langkah preprocessing:

- Mengisi nilai kosong dengan **median**
- Menstandarkan fitur dengan **StandardScaler**

📉 3. Model Baseline – Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, classification_report

logreg = LogisticRegression(max_iter=1000, class_weight="balanced", random_state=42)
pipe_lr = Pipeline([("pre", pre), ("clf", logreg)])
```

🔹 Membuat **pipeline** dengan dua langkah:

1. pre – preprocessing
2. clf – classifier (Logistic Regression)
   - 🔹 Parameter class_weight="balanced" menyeimbangkan bobot kelas minoritas.

```
pipe_lr.fit(X_train, y_train)
y_val_pred = pipe_lr.predict(X_val)
print("Baseline (LogReg) F1(val):", f1_score(y_val, y_val_pred, average="macro"))
print(classification_report(y_val, y_val_pred, digits=3))
```

◈ Melatih model dan mengevaluasi performanya pada data validasi menggunakan metrik **F1-score** dan **classification report** (precision, recall, f1, support).

🌲 4. Model Alternatif – Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    n_estimators=300, max_features="sqrt", class_weight="balanced", random_state=42
)
pipe_rf = Pipeline([("pre", pre), ("clf", rf)])

pipe_rf.fit(X_train, y_train)
y_val_rf = pipe_rf.predict(X_val)
```

◈ Membuat pipeline serupa tetapi menggunakan **Random Forest**.
◈ n_estimators=300 artinya model membuat 300 pohon keputusan.

```
pipe_rf.fit(X_train, y_train)
y_val_rf = pipe_rf.predict(X_val)
print("RandomForest F1(val):", f1_score(y_val, y_val_rf, average="macro"))
```

◈ Melatih model Random Forest dan menilai kinerjanya.

□ 5. Hyperparameter Tuning dengan GridSearchCV

```
from sklearn.model_selection import StratifiedKFold, GridSearchCV

skf = StratifiedKFold(n_splits=4, shuffle=True, random_state=42)
param = {
  "clf__max_depth": [None, 12, 20, 30],
  "clf__min_samples_split": [2, 5, 10]
}
```

◈ Membuat **cross-validation** terstratifikasi (4 fold).
◈ Mendefinisikan grid parameter yang akan dicoba untuk menemukan kombinasi terbaik.

```
gs = GridSearchCV(pipe_rf, param_grid=param, cv=skf,
                  scoring="f1_macro", n_jobs=-1, verbose=1)
gs.fit(X_train, y_train)
```

◈ Menjalankan pencarian parameter terbaik (GridSearchCV) menggunakan metrik **F1-macro**.

```
print("Best params:", gs.best_params_)
print("Best CV F1:", gs.best_score_)
```

◈ Menampilkan parameter terbaik dan hasil rata-rata skor F1 dari cross-validation.

📊 6. Evaluasi Model Terbaik

```
best_rf = gs.best_estimator_
y_val_best = best_rf.predict(X_val)
print("Best RF F1(val):", f1_score(y_val, y_val_best, average="macro"))
```

◈ Mengambil model terbaik hasil tuning dan mengevaluasinya pada data validasi.

```
from sklearn.metrics import confusion_matrix, roc_auc_score, precision_recall_curve, roc_curve
import matplotlib.pyplot as plt
```

◈ Mengimpor metrik tambahan dan plotting.

```
final_model = best_rf  # atau pipe_lr jika baseline lebih baik
y_test_pred = final_model.predict(X_test)
```

◈ Memilih model final (Random Forest atau Logistic Regression tergantung hasil terbaik).

```
print("F1(test):", f1_score(y_test,  (variable) y_test_pred: ndarray | tuple[ndarray, ndarray]
print(classification_report(y_test, y_test_pred, digits=3))
print("Confusion matrix (test):")
print(confusion_matrix(y_test, y_test_pred))
```

◈ Mengevaluasi model pada data **test**.

```
if hasattr(final_model, "predict_proba"):
    y_test_proba = final_model.predict_proba(X_test)[:,1]
    try:
        print("ROC-AUC(test):", roc_auc_score(y_test, y_test_proba))
    except:
        pass
    fpr, tpr, _ = roc_curve(y_test, y_test_proba)
    plt.figure(); plt.plot(fpr, tpr); plt.xlabel("FPR"); plt.ylabel("TPR"); plt.title("ROC (test)")
    plt.tight_layout(); plt.savefig("roc_test.png", dpi=120)
```

◈ Menghitung **ROC-AUC** dan menyimpan plot **ROC Curve** ke file roc_test.png.

💾 7. Simpan Model

```
import joblib
joblib.dump(final_model, "model.pkl")
print("Model tersimpan ke model.pkl")
```

◈ Menyimpan model terbaik ke file model.pkl agar bisa digunakan lagi tanpa melatih ulang.

🌐 8. Deploy Model dengan Flask API

```
from flask import Flask, request, jsonify
import joblib, pandas as pd

app = Flask(__name__)
MODEL = joblib.load("model.pkl")
```

🔷 Membuat aplikasi **Flask** dan memuat model yang sudah disimpan.

```python
@app.route("/predict", methods=["POST"])
def predict():
    data = request.get_json(force=True)  # dict fitur
    X = pd.DataFrame([data])
    yhat = MODEL.predict(X)[0]
    proba = None
    if hasattr(MODEL, "predict_proba"):
        proba = float(MODEL.predict_proba(X)[:,1][0])
    return jsonify({"prediction": int(yhat), "proba": proba})
```

🔷 Endpoint /predict menerima input JSON berisi fitur, lalu:

- Mengubah data menjadi DataFrame
- Melakukan prediksi (yhat)
- Jika model mendukung probabilitas (predict_proba), juga mengembalikan nilai peluang
- Hasil dikembalikan dalam format JSON

```python
if __name__ == "__main__":
    app.run(port=5000)
```

🔷 Menjalankan server Flask di port 5000.

🧠 Kesimpulan:

Kode ini merupakan **workflow lengkap pembentukan dan deployment model Machine Learning**, meliputi:

1. Pemrosesan dan pembagian dataset
2. Pembuatan pipeline dengan preprocessing otomatis
3. Pelatihan dua model (Logistic Regression & Random Forest)
4. Tuning hyperparameter menggunakan GridSearchCV
5. Evaluasi model pada data test
6. Penyimpanan model (model.pkl)
7. Deploy API prediksi dengan Flask

HASILNYA

```
..    (7, 5) (1, 5) (2, 5)
```

```
Baseline (LogReg) F1(val): 1.0
              precision    recall  f1-score   support

           1      1.000     1.000     1.000         1

    accuracy                         1.000         1
   macro avg      1.000     1.000     1.000         1
weighted avg      1.000     1.000     1.000         1
```

```
RandomForest F1(val): 1.0
```

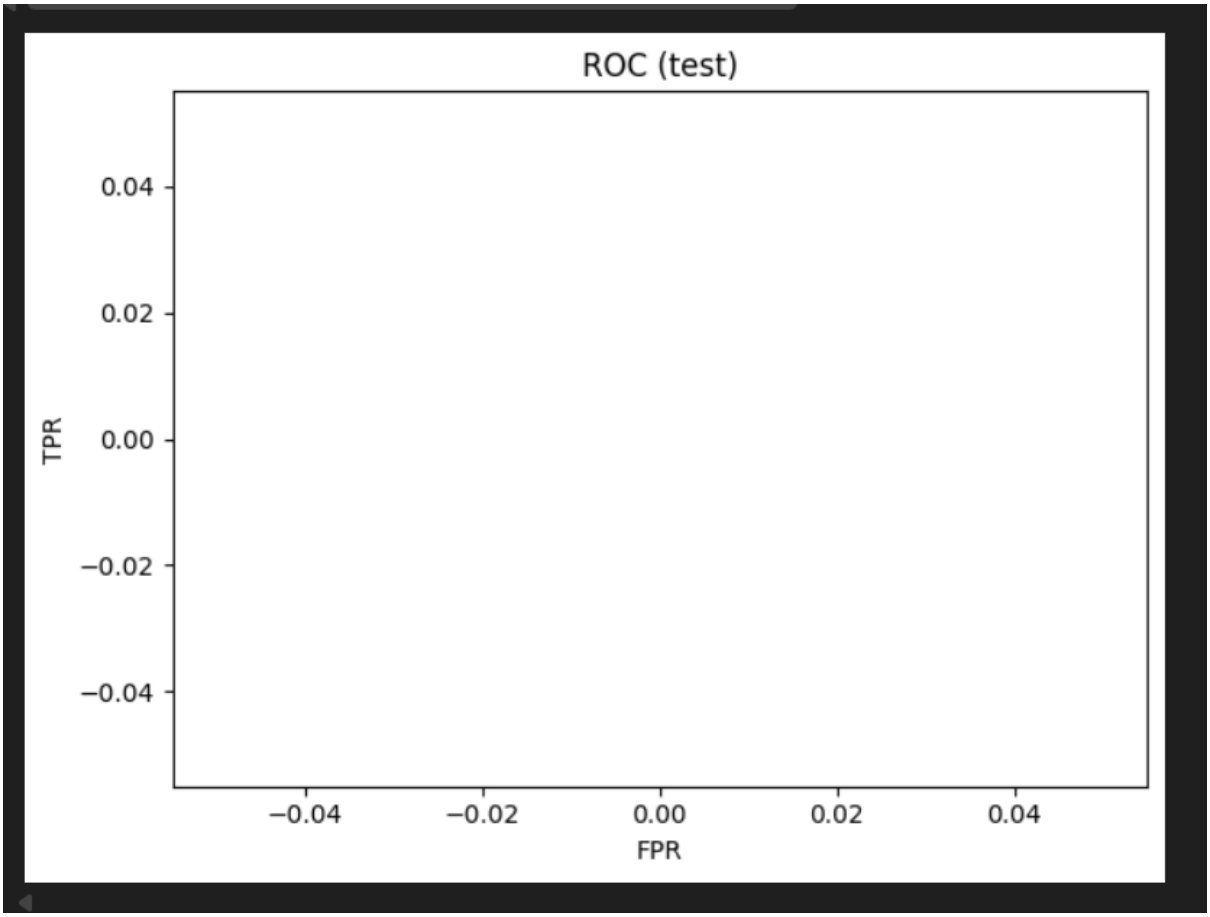
```
Fitting 4 folds for each of 12 candidates, totalling 48 fits
d:\machine_learning4\.venv\lib\site-packages\sklearn\model_selection\_split.py:811: UserWarning: The least populated cla
  warnings.warn(
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best CV F1: 0.8333333333333334
Best RF F1(val): 1.0
```

```
F1(test): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         2

    accuracy                         1.000         2
   macro avg      1.000     1.000     1.000         2
weighted avg      1.000     1.000     1.000         2

Confusion matrix (test):
[[2]]
ROC-AUC(test): nan
d:\machine_learning4\.venv\lib\site-packages\sklearn\metrics\_classification.py:534: UserWarning: A single label was fou
  warnings.warn(
d:\machine_learning4\.venv\lib\site-packages\sklearn\metrics\_ranking.py:424: UndefinedMetricWarning: Only one class is
  warnings.warn(
d:\machine_learning4\.venv\lib\site-packages\sklearn\metrics\_ranking.py:1201: UndefinedMetricWarning: No positive sampl
  warnings.warn(
```




```
Model tersimpan ke model.pkl
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```